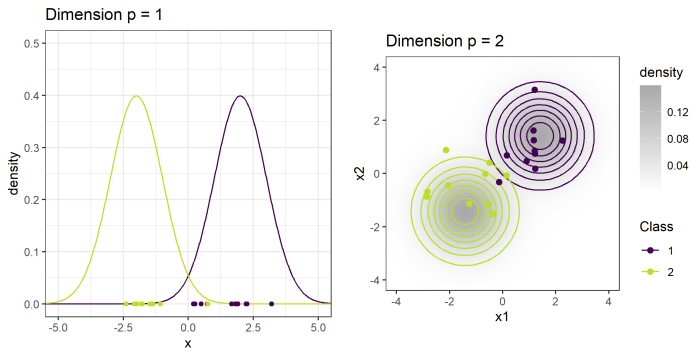
Introduction to Machine Learning

Curse of Dimensionality - Examples Learning Algorithms



Learning goals

- See how the performance of k-NN and the linear model deteriorates in high-dimensional spaces

EXAMPLE: K-NN

Let us look at the performance of algorithms for increasing dimensionality. First, we consider the k-NN algorithm:

- In a high dimensional space, data points are spread across a huge space.
- The distance to the **next neighbor** $d_{NN1}(\mathbf{x})$ becomes extremely large.
- The distance might even get so large that all points are **equally far** away - we cannot really determine the nearest neighbor anymore.

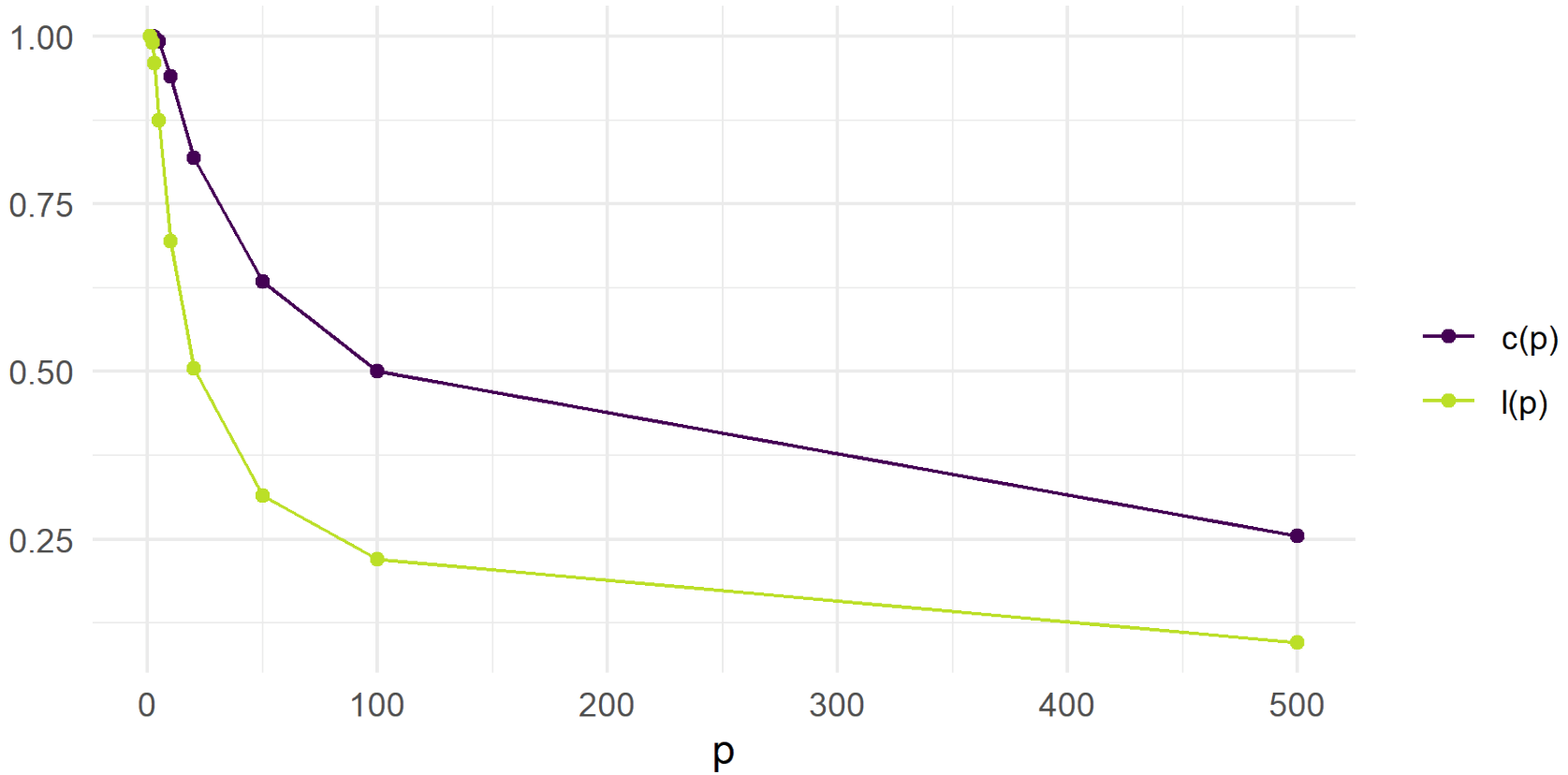
EXAMPLE: K-NN

Minimal, mean and maximal (NN)-distances of 10^4 points uniformly distributed in the hypercube $[0, 1]^p$:

p	$\min d(\mathbf{x}, \tilde{\mathbf{x}})$	$\overline{d(\mathbf{x}, \tilde{\mathbf{x}})}$	$\max d(\mathbf{x}, \tilde{\mathbf{x}})$	$\overline{d_{NN1}(\mathbf{x})}$	$\max d_{NN1}(\mathbf{x})$
1	1.2e-08	0.33	1	5e-05	0.00042
2	0.00011	0.52	1.4	0.0051	0.02
3	0.0021	0.66	1.7	0.026	0.073
5	0.016	0.88	2	0.11	0.23
10	0.15	1.3	2.5	0.39	0.63
20	0.55	1.8	3	0.9	1.2
50	1.5	2.9	4.1	2	2.4
100	2.7	4.1	5.4	3.2	3.5
500	7.8	9.1	10	8.2	8.6

EXAMPLE: K-NN

We see a decrease of relative contrast[1] $c := \frac{\max(d(\mathbf{x},\tilde{\mathbf{x}})) - \min(d(\mathbf{x},\tilde{\mathbf{x}}))}{\max(d(\mathbf{x},\tilde{\mathbf{x}}))}$ and "locality"[2] $l := \frac{\overline{d(\mathbf{x},\tilde{\mathbf{x}})} - \overline{d_{NN1}(\mathbf{x})}}{\overline{d(\mathbf{x},\tilde{\mathbf{x}})}}$ with increasing number of dimensions p:



EXAMPLE: K-NN

The consequences for the k-nearest neighbors approach can be summarized as follows:

- At constant sample size n and growing p, the distance between the observations increases
  $\rightarrow$ the coverage of the p-dimensional space decreases,
  $\rightarrow$ every point becomes isolated / far way from all other points.
- The size of the neighborhood $N_k(x)$ also "increases"
  (at constant k)
  $\rightarrow$ it is no longer a "local" method.
- Reducing k dramatically does not help much either, since the fewer observations we average, the higher the variance of our fit.

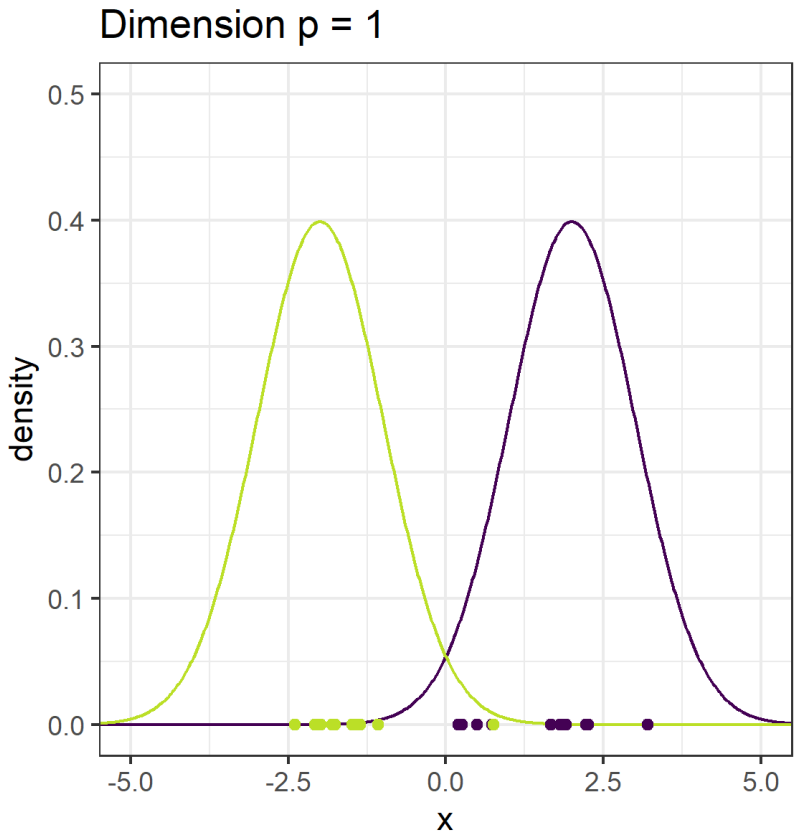
$\rightarrow$ k-NN estimates get more inaccurate with increasing dimensionality of the data.

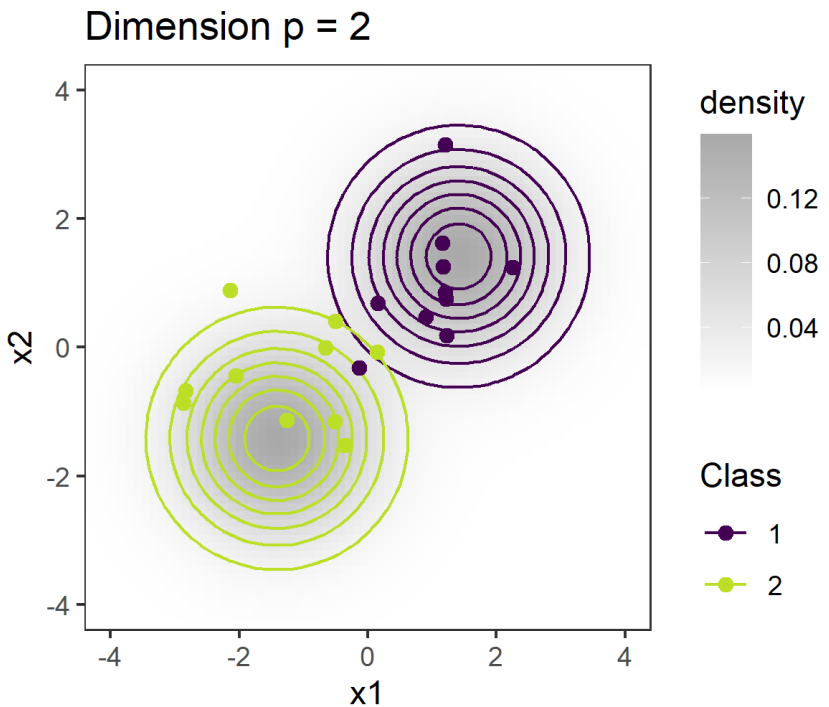
EXAMPLE: K-NN

To demonstrate this, we generate an artificial data set of dimension p as follows: We define $a = \frac{2}{\sqrt{p}}$ and

- with probability $\frac{1}{2}$ we generate a sample from class 1 by sampling from a Gaussian with mean $\boldsymbol{\mu} = (a, a, ..., a)$ and unit covariance matrix
- with probability $\frac{1}{2}$ we generate a sample from class 2 by sampling from a Gaussian with mean $-\boldsymbol{\mu} = (-a, -a, ..., -a)$ and unit covariance matrix

EXAMPLE: K-NN


Dimension p = 1
density
0.5
0.4
0.3
0.2
0.1
0.0
-5.0
-2.5
0.0
2.5
5.0
x


Dimension p = 2
x2
4
2
0
-2
-4
-4
-2
0
2
4
x1
density
0.12
0.08
0.04
Class
1
2

EXAMPLE: K-NN

This example is constructed such that the Bayes error is always constant and does not depend on the dimension p.

The Bayes optimal classifiers predicts $\hat{y} = 1$ iff

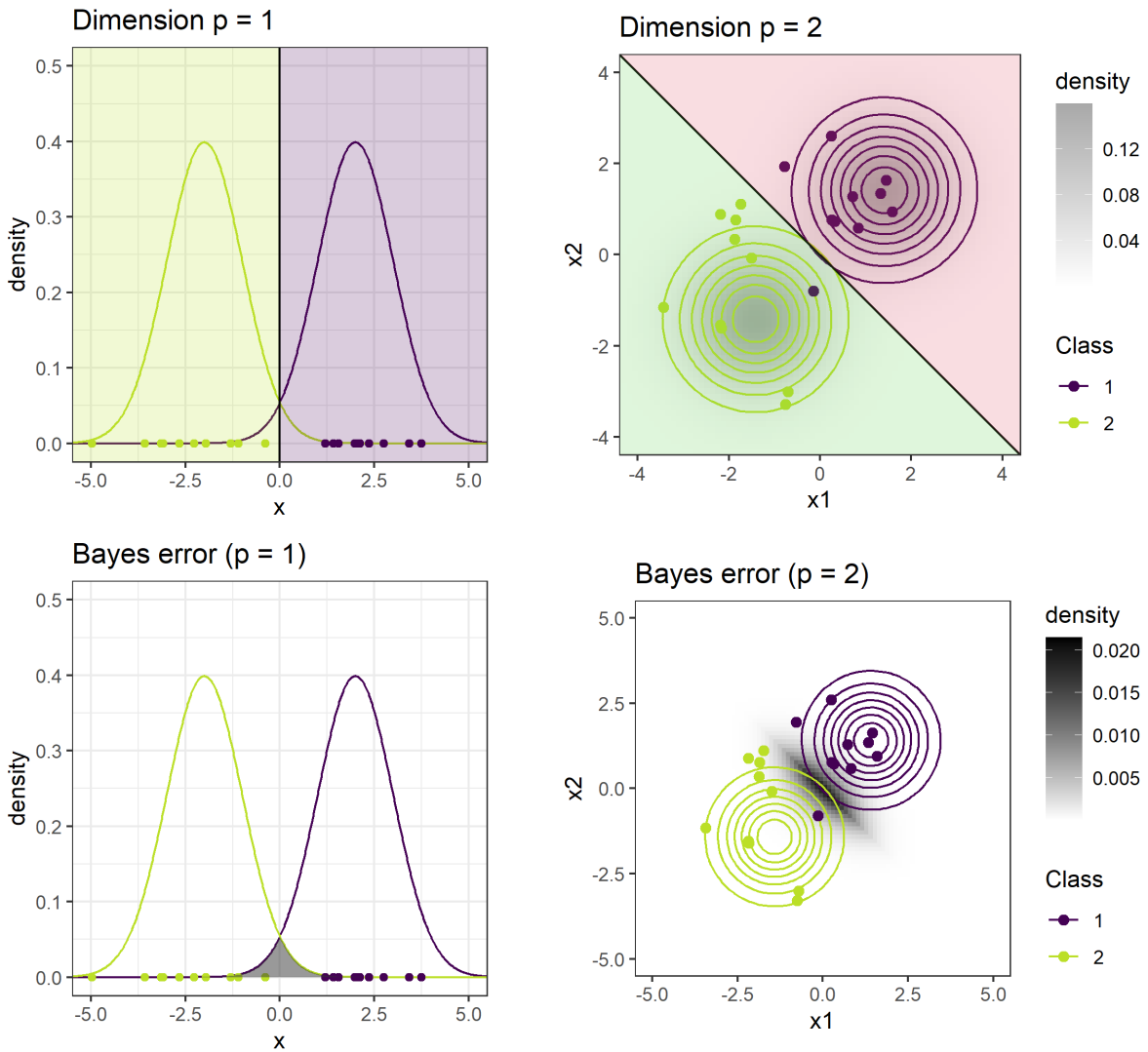
$$\begin{aligned}
\mathbb{P}(y = 1 \mid \mathbf{x}) &= \frac{p(\mathbf{x} \mid y = 1)\mathbb{P}(y = 1)}{p(\mathbf{x})} = \frac{1}{2} \cdot \frac{p(\mathbf{x} \mid y = 1)}{p(\mathbf{x})} \\
&\geq \frac{1}{2} \cdot \frac{p(\mathbf{x} \mid y = 2)}{p(\mathbf{x})} \\
&= \frac{p(\mathbf{x} \mid y = 2)\mathbb{P}(y = 2)}{p(\mathbf{x})} = \mathbb{P}(y = 2 \mid \mathbf{x}).
\end{aligned}$$

This is equivalent to

$$\begin{aligned}
\hat{y} = 1 &\Leftrightarrow \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top (\mathbf{x} - \boldsymbol{\mu})\right) \geq \exp\left(-\frac{1}{2}(\mathbf{x} + \boldsymbol{\mu})^\top (\mathbf{x} + \boldsymbol{\mu})\right) \\
&\Leftrightarrow \mathbf{x}^\top \mu \geq 0.
\end{aligned}$$

EXAMPLE: K-NN

Optimal Bayes classifier and Bayes error (shaded area):



EXAMPLE: K-NN

We can calculate the corresponding expected misclassification error (Bayes error)

$$
\begin{aligned}
& p(\hat{y} = 1 \mid y = 2)\mathbb{P}(y = 2) + p(\hat{y} = 2 \mid y = 1)\mathbb{P}(y = 1) \\
= \quad & \frac{1}{2} \cdot p(\mathbf{x}^\top \boldsymbol{\mu} \geq 0 \mid y = 2) + \frac{1}{2} \cdot p(\mathbf{x}^\top \boldsymbol{\mu} \leq 0 \mid y = 1) \\
\overset{\text{symm.}}{=} \quad & p(\mathbf{x}^\top \boldsymbol{\mu} \leq 0 \mid y = 1) = p\left(\sum_{i=1}^{p} a\mathbf{x}_i \leq 0 \mid y = 1\right) \\
= \quad & p\left(\sum_{i=1}^{p} \mathbf{x}_i \leq 0 \mid y = 1\right).
\end{aligned}
$$

$\sum_{i=1}^{p} \mathbf{x}_i \mid y = 1 \sim \mathcal{N}(p \cdot a,\ p)$, because it is the sum of independent normal random variables $\mathbf{x}_i \mid y = 1 \sim \mathcal{N}(a, 1)$ (the vector $\mathbf{x} \mid y = 1$ follows a $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I})$ distribution with $\boldsymbol{\mu} = (a, ..., a)$).

EXAMPLE: K-NN

We get for the Bayes error:

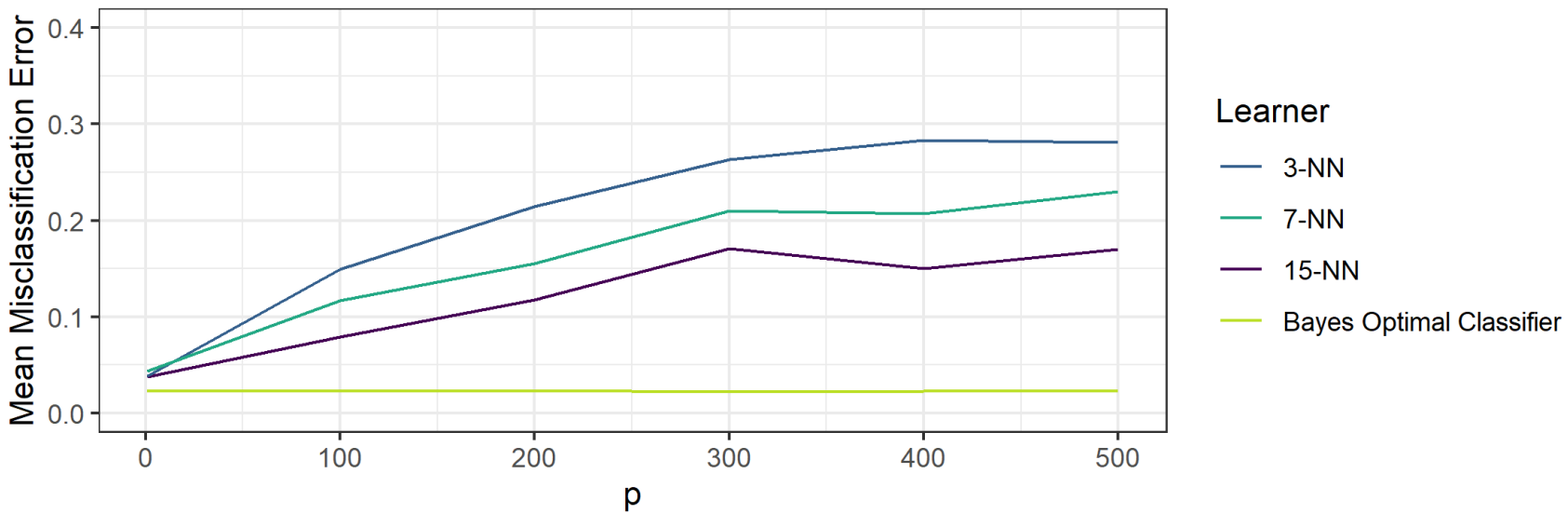
$$
\begin{aligned}
&= p\left(\frac{\sum_{i=1}^{p} \mathbf{x}_i - p \cdot a}{\sqrt{p}} \leq \frac{-p \cdot a}{\sqrt{p}} \mid y = 1\right) \\
&= \Phi(-\sqrt{p}a) \overset{a=\frac{2}{\sqrt{p}}}{=} \Phi(-2) \approx 0.0228,
\end{aligned}
$$

where Φ is the distribution function of a standard normal random variable.

We see that the Bayes error is independent of p.

EXAMPLE: K-NN

We also train a k-NN classifier for $k = 3, 7, 15$ for increasing dimensions and monitor its performance (evaluated by 10 times repeated 10-fold CV).



$\rightarrow$ k-NN deteriorates quickly with increasing dimension

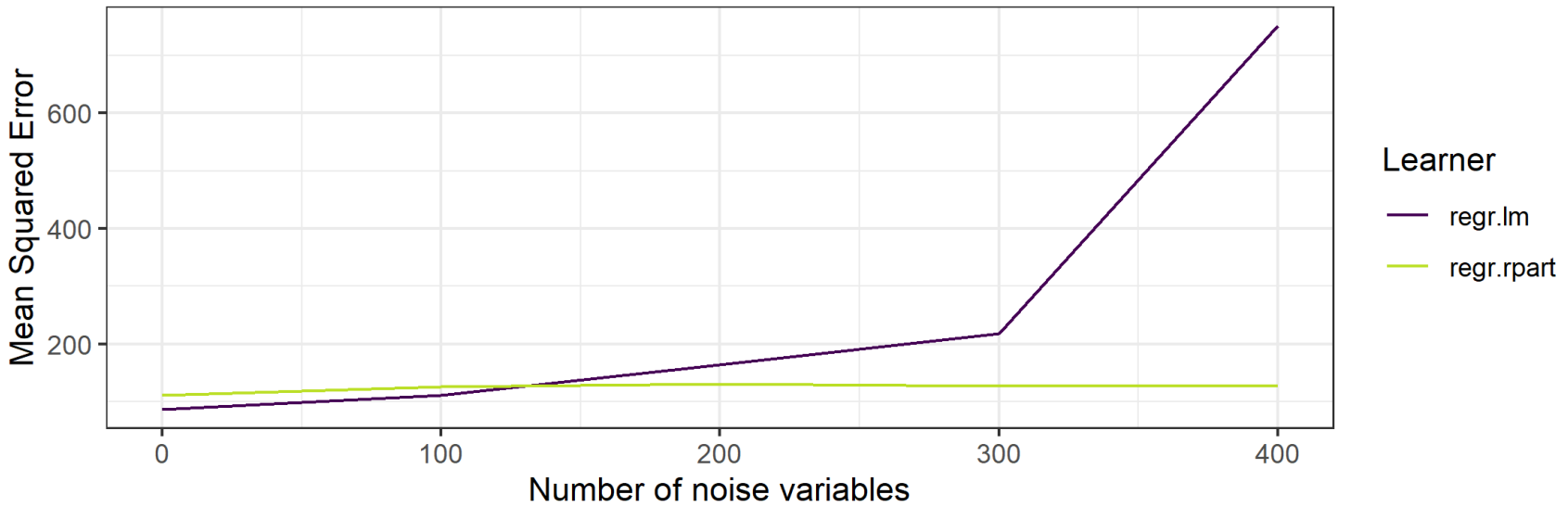
EXAMPLE: LINEAR MODEL

We also investigate how the linear model behaves in high dimensional spaces.

- We take the Boston Housing data set, where the value of houses in the area around Boston is predicted based on 13 features describing the region (e.g., crime rate, status of the population, etc.).
- We train a linear model on the data consisting of 506 observations.
- We artificially create a high-dimensional dataset by adding $100, 200, 300, \ldots$ noise variables (containing no information at all) and look at the performance of a linear model trained on this modified data (10 times repeated 10-fold CV).

EXAMPLE: LINEAR MODEL

We compare the performance of an LM to that of a regression tree.



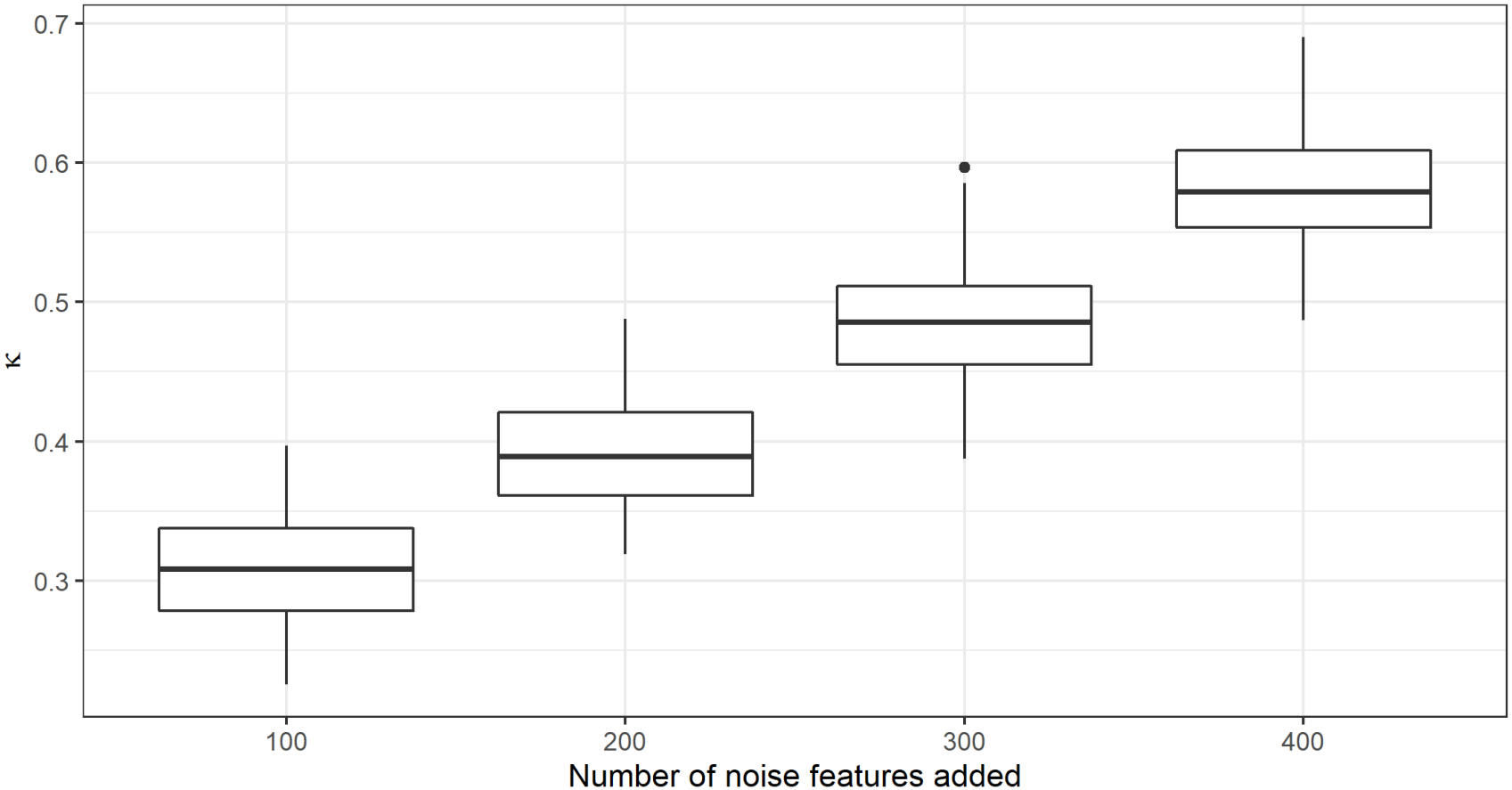
$\rightarrow$ The unregularized LM struggles with the added noise features, while our tree seems to nicely filter them out.

Note: Trees automatically perform feature selection as only one feature at a time is considered for splitting (the smaller the depth of the tree, the less features are selected). Thus, they often perform well in high-dimensional settings.

EXAMPLE: LINEAR MODEL

- The regression coefficients of the noise features can not be estimated precisely as zero in the unregularized LM due to small random correlations.
- With an increasing number of these noise features, the prediction error rises.
- To see this, we can quantify the influence of the noise features on the prediction of each observation.
  Therefore we decompose the response $\hat{y}^{(i)}$ of each iterations' test set into $\hat{y}^{(i)}_{\text{true}}$ (predicted with noise features set to 0) and $\hat{y}^{(i)}_{\text{noise}}$ (predicted with true features set to 0), s.t.
  $\hat{y}^{(i)} = \hat{y}^{(i)}_{\text{true}} + \hat{y}^{(i)}_{\text{noise}} + \text{intercept}$.
  With this, we can define the "average proportional influence of the noise features" $\kappa := \overline{\left(\frac{|\hat{y}^{(i)}_{\text{noise}}|}{|\hat{y}^{(i)}_{\text{true}}| + |\hat{y}^{(i)}_{\text{noise}}|} \right)}$.

EXAMPLE: LINEAR MODEL



When we add 400 noise features to the model, most of the time, on average, over 50% of the flexible part of the prediction ($\hat{y}^{(i)}$ − intercept) is determined by the noise features.

COD: WAYS OUT

Many methods besides k-NN struggle with the curse of dimensionality. A large part of ML is concerned with dealing with this problem and finding ways around it.

Possible approaches are:

- Increasing the space coverage by gathering more observations (not always viable in practice!)
- Reducing the number of dimensions before training (e.g. by using domain knowledge, PCA or feature selection)
- Regularization